Parsing-Based Recognition of Hierarchical Plans Using the Grammar Constraint

Kristýna Pantůčková, Simona Ondrčková, Roman Barták
Charles University, Faculty of Mathematics and Physics, Ke Karlovu 3, 121 16 Praha 2, Czech Republic
{pantuckova, ondrckova, bartak}@ktiml.mff.cuni.cz

Abstract

Plan recognition is the problem of recognizing a goal task and an agent's plan based on the observed actions. Plan recognition techniques can be employed in multi-agent systems, behaviour recognition, computer security, and other fields related to artificial intelligence. Hierarchical task networks (HTN) describe the decomposition hierarchy of tasks in planning problems. In HTN plan recognition, a prefix of the plan (actions observed so far) is given as an input, and the aim is to find a task that decomposes into a sequence of actions with the given prefix. In this paper, we show how the performance of parsing-based HTN plan recognition can be improved by restricting possible suffixes of the given prefix based on generalized arc consistency of a corresponding context-free grammar.

Introduction

Plan recognition is the task of recognizing the goal and the plan of an agent based on its action. In classical plan recognition, a sequence of observed actions of an agent is given as an input and the aim is to find the complete plan and the goal of an agent based on the knowledge of the planning domain (preconditions, effects and costs of actions), candidate goals and the initial state. Plan recognition has applications in behaviour recognition (Niu et al. 2004), computer security (Li et al. 2020), multi-agent systems (Kaminka, Pynadath, and Tambe 2002), or computer games (Ha et al. 2011).

Hierarchical planning (Erol, Hendler, and Nau 1996) deals with problems where tasks form a hierarchy. Tasks can be decomposed into subtasks until indecomposable tasks (actions), which are executable by an agent, are reached. The resulting sequence of actions is a plan of an agent. In hierarchical plan recognition, the aim is to find the root task which decomposes into an observed prefix of a plan. Hierarchical planning problems are frequently described by hierarchical task networks (HTN).

We present an improvement of an HTN plan recognition approach of (Barták, Maillard, and Cardoso 2020). We reduce the number of possible plans that the recognizer must take into account by pruning actions in an unobserved plan suffix using the grammar constraint. In this paper, we firstly mention related works, then we provide background on HTN planning recognition and grammar constrains and describe the parsing-based approach of (Barták, Maillard, and Cardoso 2020) and our approach to pruning actions in the plan suffix. Finally, we provide empirical evaluation, which shows how the grammar constraint improved the performance of parsing-based plan recognition, and conclusion.

Related works

Ramírez and Geffner (2009) proposed a plan recognition approach to plan recognition of classical (non-hierarchical) plans based on compilation to planning. Pereira, Oren, and Meneguzzi (2017) proposed an approach based on landmarks. Other approaches to classical plan recognition deal for example with on-line plan recognition (Vered et al. 2018) or epistemic plan recognition (Shvo et al. 2020).

Currently there appear to be two approaches to plan recognition in hierarchical task networks: an approach based on compilation to HTN planning (Höller et al. 2018) and an approach inspired by parsing of grammars (Barták, Maillard, and Cardoso 2020). Other hierarchical plan recognition approaches work with models weaker than HTN, e.g. (Avrahami-Zilberbrand and Kaminka 2005), (Mirsky, Gal, and Shieber 2017), or (Geib, Maraist, and Goldman 2008).

The disadvantage of the compilation-based approach (Höller et al. 2018) is that the recognizer requires a list of possible goals as an input, while the parsing-based approach (Barták, Maillard, and Cardoso 2020) does not require any additional knowledge apart from the description of the planning domain and the list of objects that can appear as parameters of tasks. However, the compilation-based approach performs better than the parsing-based approach on instances with a high number of unobserved actions as with increasing length of unobserved suffix, the number of possible combinations of actions increases exponentially. This paper addresses this problem by restricting the number of plan suffixes using the grammar constraint.

Background

HTN plan recognition

Hierarchical planning focuses on planning problems where goals (tasks) can be hierarchically decomposed into subgoals (subtasks). Indecomposable (primitive) tasks are

called actions. Actions are defined by preconditions and effects. Preconditions of an action are propositions which must be true in the state where the action is executed and effects are propositions which will be true in the state after execution of the action.

All valid decomposition of tasks into subtasks are described by methods (rules). A method describing decomposition of a task T into subtasks $T_1, ..., T_n$ corresponds to a grammar rewriting rule $T \rightarrow T_1, ...T_n[C]$, where $T_1, ..., T_n$ are either abstract or primitive tasks, C are constraints, and the order of subtasks may be arbitrary with respect to the constraints.

There are 3 types of constraints:

- $t_1 \prec t_2$ indicates that the last action from the decomposition of the task t_1 must be executed before the first action of the task t_2;
- *before*(T', p), where T' is a set of tasks and p is a proposition, indicates that p must hold in the state where the first action of the first task in the set T' is executed; and
- *between*(T', T'', p) indicates that p must be true in all states between the execution of the last action of the tasks in T' and the execution of the first action of the tasks in T''.

An HTN is described by a pair $w = (T, C)$, where T is a set of tasks and C is a set of constraints over tasks. An HTN plan recognition problem is defined as $R = (P, T, A, M, s_0, O)$, where P is a set of predicates describing states, T set of abstract tasks, A set of actions, M set of decomposition methods, s_0 is an initial state and $O =< o_1, ..., o_k >$ is an observed plan prefix of length k. The aim of plan recognition is to find a task which decomposes into a sequence of n ($k \leq n$) actions $\pi =< o_1, ..., o_k, ..., o_n >$ such that π is a valid plan applicable in s_0.

The grammar constraint

Global grammar constraints are constraints which restrict values of variables with respect to rules of a given grammar. In this paper, we implement the constraint CFG proposed by Quimper and Walsh (2006). A context-free grammar (CFG) is a formal grammar with production rules of the form $A \rightarrow \alpha$, where A is a non-terminal symbol and α is a string of terminal and non-terminal symbols.

Quimper and Walsh (2006) proposed a constraint of the form *CFG*$(G, X_1, ..., X_n)$, where $X_1...X_n$ is a string accepted by the context-free grammar G. Given domains of the variables $X_1, ..., X_n$, the authors suggest polynomial algorithms that enforce the generalized arc consistency. A constraint is generalized arc consistent (GAC) if for each value in each domain there exists a value in every other domain such that the constraint is satisfied.

The authors propose two algorithms for enforcing GAC on *CFG*$(G, X_1, ..., X_n)$. The first algorithm is a bottom-up propagator based on the algorithm CYK and the second one is a top-down propagator based on the Earley-style propagator. Both of them have the time complexity $\mathcal{O}(|G|n^3)$. The space complexity is $\mathcal{O}(|G|n^2)$ for the CYK-based algorithm and $\mathcal{O}(|G|n^3)$ for the Earley-style propagator. In our implementation, we used the Earley-style propagator as translation to the Chomsky normal form is not necessary and by the authors, top-down parsing should perform better if the CFG accepts only few strings. In addition, the authors performed experiments to compare how the efficiency of CYK and the Earley propagator depends on the number of fixed values of variables in the grammar constraint and the Earley propagator performed better than CYK on instances with more fixed variables. The Earley propagator started to outperform CYK when about half of the variables had fixed values. As we currently work with plan recognition problems with at most 1/3 unobserved actions, we expect that the Earley propagator should perform better than CYK.

Parsing-based HTN plan recognition

Parsing-based HTN plan recognition was proposed by (Barták, Maillard, and Cardoso 2020). The algorithm iteratively extends the plan length by one in each iteration and composes all abstract tasks which can be composed from the available subtasks using the available methods in the bottom-up manner, until a task which covers all observations is found.

In each iteration, the algorithm increments the plan length and attempts to put all possible actions at the new position at the end of the plan. As a consequence, the runtime of the parsing-based algorithm grows exponentially with increasing length of the unobserved plan suffix.

Parsing-based HTN plan recognition with the grammar constraint

To decrease the number of actions tried in the plan suffix and improve the performance of plan recognition with increasing suffix length, we suggest to restrict plan suffixes to those that satisfy the CFG constraint (Quimper and Walsh 2006). By omitting constraints of methods and preconditions and effects of actions, we get an abstraction of an HTN planning problem to a context-free grammar. Rules of an HTN planning domain with totally ordered subtasks correspond to production rules of a CFG. However, the constraint does not support methods with partially ordered subtasks and interleaving of actions.

The algorithm in Figure 1 describes the general idea of parsing-based HTN plan recognition using the grammar constraint. The algorithm keeps all composed subplans in the set $subplans$. Subplans store the composed task and indexes of actions from the prefix which are covered by decomposition of the task. At the beginning, the algorithm creates k subplans from the observed actions $a_1, ..., a_k$.

In each iteration, the algorithm selects all instances of decomposition methods which are applicable to the set $subplans$. A method instance is applicable to $subplans$ if all subtasks of the method are available in the set $subplans$ and composing the head task from the subtasks does not violate any constraints of the method. For each such method instance, the head task is added to the set $subplans$. If the task covers all observed actions in its decomposition, the task is

Function: RecognizePlan
Input: a sequence of observed actions $a_1, ..., a_k$
Output: a goal task g covering all observed actions
Variables: R – decomposition rules, A – available actions

```
initialize new_subplans with {a_1, ..., a_k}
for i = k, k+1, k+2, ... do
    while new_subplans ≠ ∅ do
        subplans ← subplans ∪ new_subplans
        new_subplans ← ∅
        for all rule instances applicable to subplans do
            compose new_subplan from subtasks
            if new_subplan is valid then
                if new_subplan covers a_1, ..., a_k then
                    return root task of new_subplan
                end if
                add new_subplan to new_subplans
            end if
        end for
    end while
    A_{k+1}, ..., A_i ← enforce GAC on CFG(R, X_1, ...X_i)
    where domain(X_j) = {a_j} for j ≤ k,
    domain(X_j) = A otherwise
    if A_{k+1} ≠ ∅, ..., A_i ≠ ∅ then
        initialize new_subplans with A_{k+1}, ..., A_i
    end if
end for
```

Figure 1: Parsing-based HTN plan recognition with the grammar constraint

the desired solution. Details can be found in (Barták, Maillard, and Cardoso 2020).

If the algorithm did not find any task that could cover all observations and it is not possible to create more tasks, no plan with the current length i exists. Therefore, the algorithm increases the length of a plan suffix by trying to put all possible actions at the index $i+1$. In the i-th iteration, the original parsing-based algorithm would proceed by adding all possible actions at the index $i+1$. For each possible action, a new subplan covering the index $i+1$ would be created. To decrease the number of generated subplans, we prune the suffixes using the constraint CFG.

In the constraint $CFG(R, X_1, ...X_i)$, R is a set of all methods in the form of rewriting rules without constraints, and $X_1, ..., X_i$ are variables representing actions at positions 1, ..., i in the plan. If we denote the length of the observed prefix by k, the domains of $X_1, ..., X_i$ are set as follows: $domain(X_j) = \{a_j\}$ for $1 \leq j \leq k$ and $domain(X_j) = A$ for $k < j \leq i$, where A is a set of all actions available in the plan recognition problem. The GAC propagator prunes the domains of variables $X_1, ..., X_i$. If any domain is empty, there is no valid plan for the current plan length. Otherwise, the algorithm creates a subplan covering the j-th position of the plan for each action in $domain(X_j)$ for every $k < j \leq i$.

For enforcing GAC on the constraint CFG, we used the Earley propagator. The complete algorithm for enforcing GAC on the constraint CFG can be found in (Quimper and Walsh 2006). In the following text, we describe how we used the algorithm for HTN plan recognition.

The aim of the propagator is to find a task which decomposes into the observed prefix and a suffix of the desired length. State space consists of states representing partially decomposed rules. Consider the state $s = (T_1 \rightarrow T_2...T_r \bullet T_s...T_t, j, S)$. This state represents a method which decomposes the task T_1 to the subtasks $T_2, ..., T_t$. The symbol $\bullet$ separates the subtasks that were already processed from the rest. The set S contains supports for the already decomposed subtasks, i.e., the actions in the suffix into which these subtasks decompose. At the end, the pruned domains of $X_1, ..., X_i$ will comprise the supports of all tasks that can be decomposed into a plan with the given length and prefix.

Earley parser uses 3 procedures to process states: completer, scanner, and predictor. A fully decomposed rule, e.g. $(T_1 \rightarrow T_2...T_m\bullet, j, S)$, is processed by *completer*. This state represents a possible decomposition of the task T_1 into the subtasks $T_2, ..., T_m$, which covers a subsequence of actions starting at index j. Completer will use this decomposition to decompose subtask T_1 in the rules in which subtask T_1 starting at index j is the next unprocessed subtask. A rule, where the next unprocessed subtask is an action, e.g. $(T_1 \rightarrow T_2... \bullet a_m..., j, S)$, will be processed by *scanner*. Let it be the index of the current iteration of the propagator. Then this rule already covers actions between indexes j and it, so scanner checks whether a_m is available at index $it+1$. *Predictor* processes states, where the next unprocessed subtask is an abstract task, e.g. $(T_1 \rightarrow T_2... \bullet T_m..., j, S)$. In iteration it, predictor uses all possible rules to decompose T_m into a subsequence of actions starting at index it.

Empirical evaluation

We have compared the original parsing-based approach on the domains[1] and plans[2] from the International planning competition 2020. We used totally ordered domains monroe (fully observable), transport, towers, blocksworld and satellite. All experiments were run on a computer with the Intel Core i7-11370H @ 3.30GHz processor and 16 GB of RAM. Maximum allowed runtime was limited to 15 minutes.

For a plan of length n, we created $n/3$ test instances (plan prefixes) by deleting 1 to $n/3$ actions from the end of the plan. This resulted in 351 instances for the domain monroe, 225 for satellite, 8,944 for transport, 2,734 for blocksworld, and 55,879 for towers. With the given resources, both recognizers were able to solve only simpler instances with shorter prefixes. The problems usually had increasing difficulty, so we stopped testing on a domain when the recognizers failed to solve about 100 problems in a row.

Table 1 compares the total number of problems from different domains solved by the original parsing-based recognizer and by our approach with pruning using the grammar constraint. Pruning has significantly improved the performance of parsing-based plan recognition in 3 domains: monroe, satellite and transport. In the domains blocksworld and towers, neither of the recognizers could solve more than 2 problems.

[1] https://github.com/panda-planner-dev/ipc2020-domains
[2] https://github.com/panda-planner-dev/ipc2020-plans

domain	original recognizer	with pruning
monroe	0	46
satellite	108	133
transport	13	23
blocksworld	1	2
towers	2	2

Table 1: Number of solved problems in each domain.

suffix length	1	2	3	4	5	6	7	8
solved problems	9	9	6	6	5	3	3	2

Table 2: Number of problems solved with pruning in the domain monroe.

Table 3 compares the average number of actions that were tried for problems in each domain by each recognizer until either a solution was found or until the recognizer ran out of time or memory. In the domains monroe, satellite and transport, pruning was effective in improving the performance of the recognizer. It seems that a more effective pruning algorithm could further improve the performance, especially in the domain monroe, where the number of tried actions is still very high for the unsolved problems. Although pruning could decrease the number of tried actions also in other domains, it did not result in better performance.

In the domains satellite and transport, plan recognition with pruning still did not solve any problem with unobserved suffix longer than 2. Figures 2 and 4 compare the total number of actions tried by both recognizers for the problems that were solved by both recognizers. In both domains, pruning could significantly reduce the total number of actions. Figures 3 and 5 show how pruning affected the runtimes. In the domain monroe, our approach was able to solve some problems with suffix length up to 8 (see Table 2).

Conclusion

In this paper, we have shown how parsing-based HTN plan recognition can be improved by pruning actions using the grammar constraint. Our empirical evaluation has shown that enforcing GAC on the grammar constraint for the CFG abstraction of a planning problem effectively reduces the number of actions in a plan suffix. In some of the domains, the pruning has considerably improved the performance of the recognizer.

Future work could focus on developing even more effective pruner as in some domains, the recognizer could profit from further reduction of actions. However, in other domains, pruning of actions has not led to a better performance. Therefore, it seems to be necessary to decrease the number of subplans created in each iteration in order to improve the performance of HTN plan recognition.

Acknowledgments

Research is supported by the Charles University, project GA UK number 156121 and by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215.

domain	original recognizer		with pruning	
	solved	unsolved	solved	unsolved
monroe	–	1,063,166	291	61,306
satellite	147	519	6	15
transport	605	776	43	158
blocksworld	61	89	3	44
towers	711	6,666	79	1,460

Table 3: Average number of actions tried by each recognizer until either a solution was found or until the recognizer ran out of time or memory. Considers only the cases where a recognizer reached at least the second iteration.

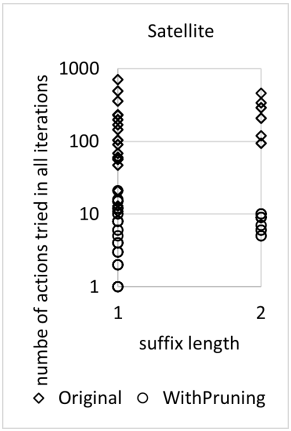

Figure 2: Number of new actions created by each recognizer in the domain satellite (logarithmic scale).

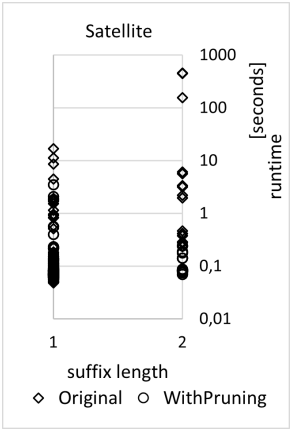

Figure 3: Runtimes of each recognizer in the domain satellite (logarithmic scale).

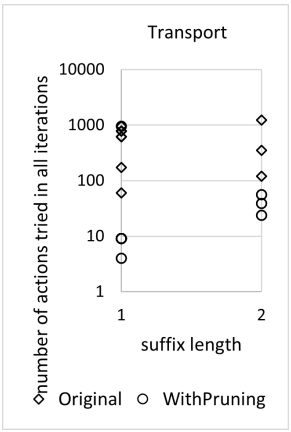

Figure 4: Number of new actions created by each recognizer in the domain transport (logarithmic scale).

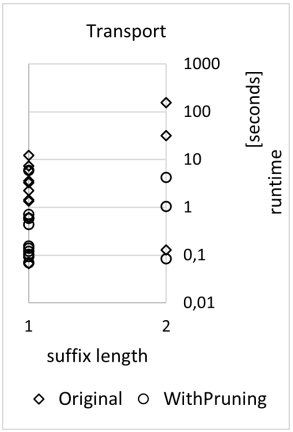

Figure 5: Runtimes of each recognizer in the domain transport (logarithmic scale).

References

Avrahami-Zilberbrand, D., and Kaminka, G. A. 2005. Fast and complete symbolic plan recognition. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 653–658.

Barták, R.; Maillard, A.; and Cardoso, R. C. 2020. Parsing-based approaches for verification and recognition of hierarchical plans. In *The AAAI 2020 Workshop on Plan, Activity, and Intent Recognition*.

Erol, K.; Hendler, J. A.; and Nau, D. S. 1996. Complexity Results for HTN Planning. *Annals of Mathematics and AI* 18(1):69–93.

Geib, C. W.; Maraist, J.; and Goldman, R. P. 2008. A new probabilistic plan recognition algorithm based on string rewriting. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling*, 91–98.

Ha, E.; Rowe, J.; Mott, B.; and Lester, J. 2011. Goal recognition with markov logic networks for player-adaptive games. In *Proceedings of the seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 6.

Höller, D.; Behnke, G.; Bercher, P.; and Biundo, S. 2018. Plan and goal recognition as HTN planning. In *2018 IEEE 30th International Conference on Tools with Artificial Intelligence*, 466–473.

Kaminka, G. A.; Pynadath, D. V.; and Tambe, M. 2002. Monitoring teams by overhearing: A multi-agent plan-recognition approach. *Journal of artificial intelligence research* 17:83–135.

Li, T.; Liu, Y.; Liu, Y.; Xiao, Y.; and Nguyen, N. A. 2020. Attack plan recognition using hidden Markov and probabilistic inference. *Computers & Security* 97:101974.

Mirsky, R.; Gal, Y.; and Shieber, S. M. 2017. CRADLE: an online plan recognition algorithm for exploratory domains. *ACM Transactions on Intelligent Systems and Technology* 8(3):1–22.

Niu, W.; Long, J.; Han, D.; and Wang, Y.-F. 2004. Human activity detection and recognition for video surveillance. In *Proceedings of the 2004 IEEE International Conference on Multimedia and Expo (ICME)*, volume 1, 719–722.

Pereira, R. F.; Oren, N.; and Meneguzzi, F. 2017. Landmark-based heuristics for goal recognition. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 3622–3628.

Quimper, C.-G., and Walsh, T. 2006. Global grammar constraints. In *Principles and Practice of Constraint Programming-CP 2006: 12th International Conference, CP 2006, Nantes, France, September 25-29, 2006. Proceedings 12*, 751–755. Springer.

Ramírez, M., and Geffner, H. 2009. Plan recognition as planning. In *Proceedings of the twenty-first International Joint Conference on Artifical Intelligence*, 1778–1783.

Shvo, M.; Klassen, T. Q.; Sohrabi, S.; and McIlraith, S. A. 2020. Epistemic plan recognition. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 1251–1259.

Vered, M.; Pereira, R. F.; Kaminka, G.; and Meneguzzi, F. R. 2018. Towards online goal recognition combining goal mirroring and landmarks. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*, 2112–2114.